Approximating Rate-Distortion Graphs of Individual Data: Experiments in Lossy Compression and Denoising

Steven de Rooij and Paul M.B. Vitányi

Abstract—Classical rate-distortion theory requires specifying a source distribution. Instead, we analyze rate-distortion properties of individual objects using the recently developed algorithmic rate-distortion theory. The latter is based on the noncomputable notion of Kolmogorov complexity. To apply the theory we approximate the Kolmogorov complexity by standard data compression techniques, and perform a number of experiments with lossy compression and denoising of objects from different domains. We also introduce a natural generalization to lossy compression with side information. To maintain full generality we need to address a difficult searching problem. While our solutions are therefore not time efficient, we do observe good denoising and compression performance.

Index Terms—Compression, denoising, rate-distortion, structure function, Kolmogorov complexity.

1 Introduction

Rate-distortion theory analyzes communication over a channel under a constraint on the number of transmitted bits, the "rate." It currently serves as the theoretical frame of reference for many important applications such as lossy compression; it can also be applied to denoising, and more generally, applications that require a separation of structure and noise in the input data.

Classical rate-distortion theory evolved from Shannon's theory of communication [1]. It describes the trade-off between the rate and the achievable fidelity of the transmitted representation under some distortion function, where the analysis is carried out *in expectation* under some source distribution. Therefore the theory can only be meaningfully applied if we have some inkling as to the distribution on objects that we want to compress lossily. While lossy compression is ubiquitous, propositions with regard to the underlying distribution tend to be ad hoc, because the assumption that the objects of interest are drawn from a single distribution is questionable. Moreover, even if a true source distribution is known to exist, it is typically hard to determine what it is like, and objects that occur in practice all too often exhibit more structure than expected under the source model of choice.

For large outcome spaces then, it becomes important to consider structural properties of *individual objects*. For example, if the rate is low, then we may still be able to transmit objects that have a very regular structure without introducing any distortion, but this becomes impossible for objects with high information density. In his 1980 paper [2], Ziv shows that there is a universal algorithm that *asymptotically* performs as well as the optimal finite state method for a fixed but infinite individual sequence. As such it is not necessary to specify a source distribution. For more modern (and practically useful) research along those lines see, e.g., [3], [4].

In order to define a *nonasymptotic* rate-distortion theory that allows analysis of individual finite objects, the notion of Kolmogorov complexity is required. Donoho made the first foray in this direction [5]; the present work is inspired by [6], which lays down the foundations for an algorithmic analogue of Shannon's probabilistic rate-distortion theory. There is a problem: although this theory allows for an elegant formalization of rate-distortion for individual objects, it cannot be applied directly, as Kolmogorov complexity is uncomputable.

Our aim is to test this new theory in practice. We approximate Kolmogorov complexity by the compressed size of the object using a (lossless) general purpose data compression algorithm, and conduct a number of experiments in two distinct applications of the theory, namely lossy compression and denoising. A number of studies have pointed out the relationship between lossy compression and denoising; see, e.g., [7], but the present framework makes it particularly easy to explain the connection.

In Section 2, we briefly introduce algorithmic rate-distortion theory. We also generalize the theory to settings with side information. A practical version of the theory is outlined in Section 3. Finding the approximate rate-distortion function is a difficult search problem. We motivate and outline the genetic algorithm we used to approximate the rate-distortion function. Then, in Section 4, we describe five experiments in lossy compression and denoising. The results are presented and discussed in Section 5. In Section 6, we take a step back and discuss to what extent our practical approach yields a faithful approximation of the theoretical algorithmic rate-distortion function. In Section 7, we relate our framework to a Minimum Description Length approach. Our findings are summarized in Section 8.

• *The authors are with the Centrum Wiskunde & Informatica (CWI), Kruislaan 413, PO Box 94079, Amsterdam 1090 GB, The Netherlands. E-mail: {s.de.rooij, S.de.Rooij}@cwi.nl.*

Manuscript received 1 Sept. 2010; accepted 5 Dec. 2010; published online 1 Feb. 2010.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2010-09-0485.
Digital Object Identifier no. 10.1109/TC.2011.25.

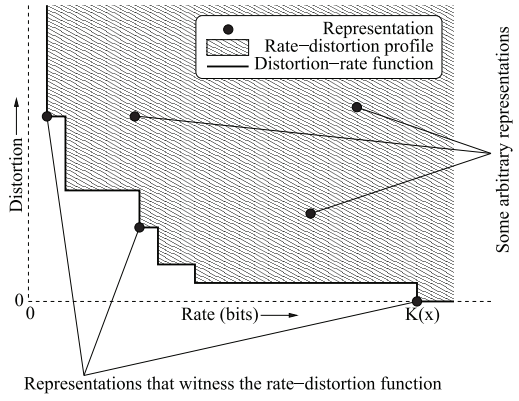


Fig. 1. Rate-distortion profile and distortion-rate function.

2 Algorithmic Rate-Distortion Theory

This section very briefly summarizes the relevant theory in [6]. Suppose we want to communicate objects x from a countable set of source words $\mathcal{X}$ using at most r bits per object. We call r the *rate*. We locate a good *representation* of x within a finite set $\mathcal{Y}$, which may be different from $\mathcal{X}$ in general (but we usually have $\mathcal{X} = \mathcal{Y}$ in this text). The lack of fidelity of a representation y is quantified by a distortion function $d : \mathcal{X} \times \mathcal{Y} \to [0, \infty)$.

The Kolmogorov complexity of y, denoted $K(y)$, is the length of the shortest program that constructs y. More precisely, it is the length of the shortest input to a fixed universal binary prefix machine that will output y and then halt; also see the textbook [8]. We can transmit that shortest program for any representation y that has $K(y) \leq r$; the receiver can then run the program to obtain y and is thus able to reconstruct x up to distortion $d(x, y)$. Define the *rate-distortion profile* P_x *of the source word* x as the set of pairs $\langle r, a \rangle$ such that there is a representation $y \in \mathcal{Y}$ with $d(x, y) \leq a$ and $K(y) \leq r$. The possible combinations of r and a can also be characterized by the *rate-distortion function of the source word* x, which is defined as $r_x(a) = \min\{r \mid \langle r, a \rangle \in P_x\}$, or by the *distortion-rate function of the source word* x, which is defined as $d_x(r) = \min\{a \mid \langle r, a \rangle \in P_x\}$. These two functions are somewhat like inverses of each other; although strictly speaking they are not since they are monotonic but not strictly monotonic. Also note that, unlike in classical rate-distortion theory, *each source word* has associated rate-distortion and distortion-rate functions. A representation y is said to *witness* the rate-distortion function of x if $r_x(d(x, y)) = K(y)$. These definitions are illustrated in Fig. 1.

Algorithmic rate-distortion theory is a generalization of Kolmogorov's structure function theory. It is developed and treated in much more detail in [6].

2.1 Side Information

Suppose we want to transmit a source word $x \in \mathcal{X}$ and we have chosen a representation $y \in \mathcal{Y}$ as before. The encoder and decoder often share a lot of information: both might know that grass is green and the sky is blue, they might share a common language, and so on. They would not need to transmit such information. It seems reasonable to allow the program to compute a representation y, transmitted from an encoder to a decoder, to use any side information z they might share. Such programs may be shorter than they could be otherwise. This can be formalized by switching to the *conditional* Kolmogorov complexity $K(y|z)$, which is the length of the shortest Turing machine program that constructs y on input z. We redefine $K(y) = K(y|\epsilon)$, where ϵ is the empty sequence, so that $K(y|z) \leq K(y)$ up to an independent constant: the length of the shortest program for y can never significantly increase when side information is provided, but it might certainly decrease when y and z share a lot of information [8]. We change the definitions as follows: the *rate-distortion profile of the source word* x *with side information* z is the set of pairs $\langle r, a \rangle$ such that there is a representation $y \in \mathcal{Y}$ with $d(x, y) \leq a$ and $K(y|z) \leq r$. The definitions of the rate-distortion function and the distortion-rate function are similarly changed. In Section 5, we will demonstrate an application: removal of errors in short text documents. Henceforth, we will omit mention of the side information z unless it is relevant to the discussion.

2.2 Distortion Spheres, the Minimal Sufficient Statistic

A representation y that witnesses the rate-distortion function is the best possible rendering of the source object x because it achieves a distortion of $d(x, y)$ at the lowest possible rate, but if the rate is lower than $K(x)$, then some information is necessarily lost. As we want to find the best possible separation between structure and noise in the data, it is important to determine to what extent the discarded information is noise.

Together, a representation y and the distortion $a = d(x, y)$ conveys the information that the source object x can be found somewhere on the list of all $x' \in \mathcal{X}$ that satisfy $d(x', y) = a$. We call such a list a *distortion sphere*. A distortion sphere of radius a, centered around y is defined as follows:

$$S_y(a) = \{x' \in \mathcal{X} : d(x', y) = a\}. \tag{1}$$

If x is a completely random element of this list, then the discarded information is pure "white noise": it contains no meaningful information. Conversely, all random elements in the list share all "simply described" (in the sense of having low Kolmogorov complexity) properties that x satisfies. Hence, with respect to the "simply described" properties, every such random element is as good as x, see [6] for more details. In such cases, given that x is in $S_y(a)$, a literal specification of the index of x in the list is the most efficient code for x. A fixed-length, literal code requires $\log |S_y(a)|$ bits. (Here and in the following, all logarithms are taken to base 2.) On the other hand, if the discarded information is structured, then the Kolmogorov complexity of the index of x in $S_y(a)$ will be significantly lower than the logarithm of the size of the sphere. The difference between these two code lengths can be used as an indicator of the amount of structural information that is discarded by the representation y. Vereshchagin and Vitányi [6] call this quantity the *randomness deficiency* of the source object x in the set $S_y(a)$, and they show that if y witnesses the rate-distortion function of x, then it *minimizes* the randomness deficiency at rate $K(y)$; thus the rate-distortion function

identifies those representations that account for as much structure as possible at the given rate.

To assess how much structure is being discarded at a given rate, consider a code for the source object x in which we first transmit the shortest possible program that constructs y, then the shortest possible program that constructs the radius of the distortion sphere, and finally a literal, fixed-length index of x in the distortion sphere $S_y(a)$. Such a code has the following length function:

$$L_y(x) = K(y) + K(d(x,y)|y) + \log|S_y(d(x,y))|. \quad (2)$$

It is possible that $d(x,y)$ is fully determined by y, for example if $\mathcal{Y}$ is the set of all finite subsets of $\mathcal{X}$ and list decoding distortion is used, as described in [6]. In such cases, $K(d(x,y)|y)$ never exceeds a fixed constant. However, this does not apply in our experiments; we actually need to encode $d(x,y)$ separately. However, the number of bits required is negligible compared to the total three part code length.

If the rate is very low then the representation y models only very basic structure and the randomness deficiency in the distortion sphere around y is high. Borrowing terminology from statistics, we may say that y is a representation that "underfits" the data: y does not capture all relevant features of the data. In such cases, we should find that $L_y(x) > K(x)$, because the fixed-length code for the index of x within the distortion sphere is suboptimal in this case. But suppose that y is complex enough that it satisfies $L_y(x) \approx K(x)$ (making this definition more precise is outside the scope of this paper). In [6], such representations are called *(algorithmic) sufficient statistics* for the data x. A sufficient statistic has close to zero randomness deficiency, which means that it represents all structure that can be detected in the data. However, sufficient statistics might contain not only structure, but noise as well. Such a representation would be overly complex, an example of overfitting. A *minimal* sufficient statistic balances between underfitting and overfitting. It is defined as the lowest complexity sufficient statistic, which is the same as the lowest complexity representation y that minimizes the code length $L_y(x)$. As such it can also be regarded as the "model" that should be selected on the basis of the Minimum Description Length (MDL) principle; also see Section 7. To be able to relate the distortion-rate function to this code length we define the *code length function* $\lambda_x(r) = L_y(x)$ where y is the representation that minimizes the distortion at rate r.

2.3 Applications: Denoising and Lossy Compression

Representations that witness the rate-distortion function provide optimal separation between structure that can be expressed at the given rate and residual information that is perceived as noise. Therefore, these representations can be interpreted as denoised versions of the original. Since the minimal sufficient statistic discards as much noise as possible, without losing any structure, it is the best candidate for applications of denoising.

While the minimal sufficient statistic is a denoised representation of the original signal, it is not necessarily given in a directly usable form. For instance, $\mathcal{Y}$ could consist of subsets of $\mathcal{X}$, but a *set* of source words is not always acceptable as a denoising result. So in general, one may need to apply some function $f : \mathcal{Y} \to \mathcal{X}$ to the sufficient statistic to construct a usable object. But if $\mathcal{X} = \mathcal{Y}$ and the distortion function is a metric, as in our case, then the representations are already in an acceptable format, so here we use the identity function for the transformation f.

In applications of lossy compression, one may be willing to accept a rate which is lower than the complexity $K(y)$ for a minimal sufficient statistic y, thereby losing some structural information. However, theory does tell us that it is not worthwhile to set the rate to a value higher than $K(y)$. The original object x is a random element of $S_y(d(x,y))$, and it cannot be distinguished from any other random $z \in S_y(d(x,y))$ using only "simply described" properties. So we have no "simply described" test to discredit the hypothesis that any such z is the original object, given y and $d(x,y)$. But increasing the rate, yielding a model y' and $d(x,y') < d(x,y)$, we commonly obtain a sphere $S_{y'}$ of smaller cardinality than S_y, with some random elements of S_y not being random elements of $S_{y'}$. Since these excluded elements were perfectly good candidates of being the original object, if the rate is higher than $K(y)$, the resulting representation y' models irrelevant features ("noise") that are specific to x: the representation starts to "overfit."

In lossy compression, as in denoising, the representations themselves may be unsuitable for presentation to the user. For example, when decompressing a lossily compressed image, in most applications a *set* of images would not be an acceptable result. So again a transformation from representations to objects of a usable form has to be specified. There are two obvious ways of doing this. First, if a representation y witnesses the rate-distortion function for a source word $x \in \mathcal{X}$, then any two random objects in $S_y(d(x,y))$ cannot be distinguished from one another at rate $K(y)$. Therefore, we might choose not to use a deterministic transformation, but rather report the uniform distribution on the objects in $S_y(d(x,y))$ as the lossily compressed version of x. This method has the advantage that it is applicable whether or not $\mathcal{X} = \mathcal{Y}$. Second, if $\mathcal{X} = \mathcal{Y}$ and the distortion function is a metric, then it makes sense to use the identity transformation again, although here the motivation is different. Suppose we select some $x' \in S_y(d(x,y))$ other than y. Then the best upper bound we can give on the distortion is $d(x,x') \le d(x,y) + d(y,x') = 2d(x,y)$ (by the triangle inequality and symmetry). Thus, the distortion for y is only half of the upper bound on the distortion we obtained for x'. Therefore, using y is more suitable from a worst-case perspective. This method has as an additional advantage that the decoder does not need to *know* the distortion $d(x,y)$ which often cannot be computed from y without knowledge of x.

Of these two approaches, if the rate is high enough to transmit a sufficient statistic, the first seems preferable. We have nevertheless chosen to always report y directly in our analysis, which has the advantage that this way, all reported results are of the same type.

3 Computing Individual Object Rate Distortion

The rate-distortion function for an object x with side information z and a distortion function d is found by simultaneously minimizing two objective functions

$$g_1(y) = K(y|z) \text{ and } g_2(y) = d(x, y).$$

We impose a partial order on representations

$$y \preceq y', \text{ iff } g_1(y) \leq g_1(y') \text{ and } g_2(y) \leq g_2(y'). \quad (3)$$

Our goal is to find the set of representations that are minimal under $\preceq$. Such an optimization problem cannot be implemented because of the uncomputability of K. To make the idea practical, we need to approximate the conditional Kolmogorov complexity. As observed in [9], it follows directly from symmetry of information for Kolmogorov complexity (see [8, p. 233]) that

$$K(y|z) = K(zy) - K(z) + O(\log n), \quad (4)$$

where n is the length of zy. To approximate conditional complexity, we will ignore the logarithmic term, and replace K by $\tilde{K}$, the length of the compressed representation under a general purpose compression algorithm.[1] The approximate conditional complexity thus becomes

$$\tilde{K}(y|z) = \tilde{K}(zy) - \tilde{K}(z) \approx K(y|z), \quad (5)$$

and the definition of g_1 is changed to use $\tilde{K}$ rather than K. This may be a poor approximation: $\tilde{K}(y)$ may be quite high even for objects y that have $K(y)$ close to zero. Our results show evidence that some of the theoretical properties of the distortion-rate function nevertheless carry over to the practical setting; we also explain how some observations that are not predicted by theory are in fact related to the (unavoidable) inefficiencies of the used compressor.

Compressor (rate function). One can use any general-purpose compressor in (5); in our experiments we used a block sorting compression algorithm with a move-to-front scheme as described in [10]. The algorithm is very similar to a number of common general purpose compressors, such as bzip2 and zzip [11], but it is simpler and faster for small inputs; the source code (in C) is available from the authors.

Of course, domain specific compressors might yield better compression for some object types (such as sound wave files), and therefore a better approximation of the Kolmogorov complexity. However, we could find no standard algorithms that substantially outperform ours on the very small inputs that we experimented with.

Code length function. In Section 2.2, we introduced the code length function $\lambda_x(r)$. Its definition makes use of (2), for which we have not yet provided a computable alternative. We use the following approximation:

$$\tilde{L}_y(x) = \tilde{K}(y) + L_D(d(x, y)|y) + \log |S_y(d(x, y))|. \quad (6)$$

In this paper we use an Elias code [12] for L_D, with code length $L_D(d) = \log(d + 1) + 2 \log \log(d + 2) + O(1)$.

Distortion functions. In all our experiments (Section 4), we have $\mathcal{X} = \mathcal{Y}$ and the used distortion functions are metrics. We use the following three common metrics:

1. *Hamming distortion.* Hamming distortion is perhaps the simplest distortion function that could be used. Let x and y be two objects of equal length n. The Hamming distortion $d(x, y)$ is equal to the number of symbols in x that do not match those in the corresponding positions in y.
2. *Euclidean distortion.* As before, let $x = x_1, \ldots, x_n$ and $y = y_1, \ldots, y_n$ be two objects of equal length, but the symbols now have a numerical interpretation. Euclidean distortion is

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} :$$

the distance between x and y when they are interpreted as vectors in an n-dimensional euclidean space. In other words, this distortion is the root of the summed squared error. Note that this definition of euclidean distortion differs from the one in [6].
3. *Edit distortion.* The edit distortion, or Levenshtein distortion, is a well-known criterion for approximate string matching [13]. The edit distortion of two strings x and y, possibly of different lengths, is the minimum number of symbols that have to be deleted from, inserted into, or changed in x in order to obtain y (or vice versa).

3.1 Searching for the Rate-Distortion Function

The search problem that we have to address is hard for three reasons. First, the search space is very large: for an object of n bits there are 2^n candidate representations of the same size, and objects that are typically subjected to lossy compression are often millions or billions of bits long. Thus, an exhaustive search algorithm is not practical. Second, we found on the other hand that a greedy search procedure tends to terminate quickly in some local optimum that is very bad globally. Third, we want to avoid making too many assumptions about the two objective functions, so that we are free to change the compression algorithm and the distortion function.

Since the structure of the search landscape is at present poorly understood and we do not want to make any unjustifiable assumptions, we use a genetic search algorithm which performs well enough that interesting results can be obtained. More specialized Monte Carlo algorithms may yield faster performance; one such approach is described in [4].

We need a number of definitions for the subsequent discussion. A finite subset of $\mathcal{Y}$ is called a *pool*. A pool $\mathcal{P}$ induces a *trade-off profile* $p(\mathcal{P}) = \{\langle g_1(y), g_2(y)\rangle \mid y \in \mathcal{Y}, y' \preceq y \text{ for some } y' \text{ in } \mathcal{P}\}$. The *weakness* $w_{\mathcal{P}}(y)$ of an object $y \in \mathcal{P}$ is defined as the number of elements of the pool that are smaller according to $\preceq$. The *(transitive) reduction* $\mathrm{trd}(\mathcal{P})$ of a pool $\mathcal{P}$ is the subset of all elements with zero weakness. Elements of $\mathrm{trd}(\mathcal{P})$ are also called *models*.

The search algorithm initializes a pool $\mathcal{P}_0$, which is then subjected to a process of selection through survival of the fittest: the pool is iteratively updated by replacing elements with low fitness by new ones, which are created through either *mutation* (random modifications of elements) or *crossover* ("genetic" recombination of pairs of other candidates). We write $\mathcal{P}_i$ to denote the pool after i iterations. When the algorithm terminates after n iterations it outputs the reduction of $\mathcal{P}_n$. Implementation details are in the online appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TC.2011.25.

1. The logarithmic overhead can be avoided by approximating conditional complexity directly using a sequential compressor; see the end of Section 4.

4 Experiments

We subjected five objects to our program. The following considerations have guided our choice of objects:

1. Objects should not be too complex, allowing our program to find a good approximation of the distortion-rate curve. We found that the running time of the program seems to depend mostly on the complexity of the original object; a compressed size of 20,000 bits seemed to be about the maximum our program could handle within a reasonable amount of time, requiring a running time of the order of weeks on a desktop computer.
2. To check that our method really is general, as much as possible we selected objects from different domains, for which different distortion functions are appropriate. Preferably, the objects contain structure at different levels of complexity.
3. Objects should contain primary structure and regularities that are distinguishable and compressible by a block sorting compressor such as the one we use. Otherwise, the assumption that compressor implements a reasonable approximation of Kolmogorov complexity becomes very strained. For instance, we would not expect our program to do well on a sequence of digits from the binary expansion of the number π.

With this in mind, we have selected the objects shown in Fig. 2.

In each experiment, as time progressed the program found less and less improvements per iteration, but the pool never stabilized completely. Therefore we interrupted each experiment when 1) after at least one night of computation, the pool did not improve a lot, and 2) for all intuitively good models $y \in \mathcal{Y}$ that we could conceive of a priori, the algorithm had found an y' in the pool with $y' \preceq y$ according to (3). In each denoising experiment, this test included the original, noiseless object. In the experiment, on the mouse without added noise, we also included the images that can be obtained by reducing the number of gray levels in the original with an image manipulation program. Finally, for the grayscale images we included a number of objects that can be obtained by subjecting the original object to JPEG compression at various quality levels.

The first experiment illustrates how algorithmic rate-distortion theory may be applied to lossy compression problems, and it illustrates how for a given rate, some features of the image are preserved while others can no longer be retained. We compare the performance of our method to the performance of JPEG (the 2000 standard). JPEG images were encoded to jpc format with three quality levels using NetPBM version 10.33.0; all other options are default. For more information about this software refer to [14].

The other four experiments are concerned with denoising. Any model that is output by the program can be interpreted as a denoised version of the input object. We measure the denoising success of a model y as $d(x', y)$, where x' is the original version of the input object x, before noise was added. We also compare the denoising results to those of other denoising algorithms:

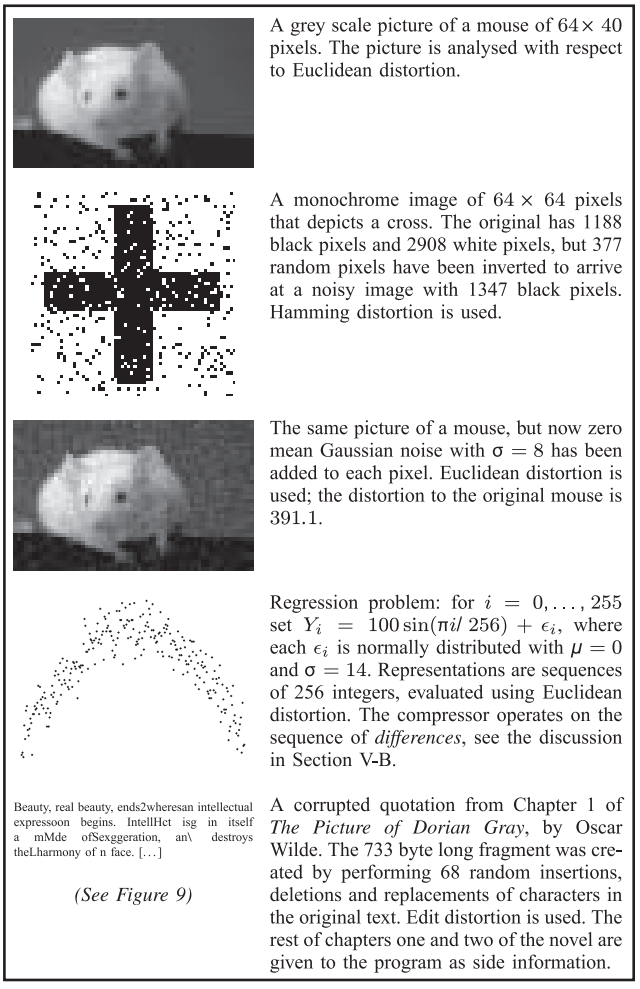


Fig. 2. Objects of experimentation.

1. Blurring (convolution with a Gaussian kernel). Blurring works like a low-pass filter, eliminating all high frequency information including noise. Other high frequency features of the image, such as sharp contours, are also discarded.
2. Naive denoising. We applied a naive denoising algorithm to the noisy cross, in which each pixel was inverted if five or more out of the eight neighboring pixels were of different color.
3. Denoising based on JPEG. Here we subjected the noisy input image to JPEG compression at various quality levels. We then selected the result for which the distortion to the original image was lowest.

We have also tried BayesShrink [15], a more sophisticated, wavelet-based denoising algorithm. However it turns out that BayesShrink hardly affects the input image at all, probably because it is so small.

4.1 Names of Objects

To facilitate description and discussion of the experiments we will adopt the following naming convention. Objects related to the experiments with the mouse, the noisy cross, the noisy mouse, the sine wave, and the Wilde fragment, are denoted by the symbols $\mathbb{M}$, $\mathbb{C}$, $\mathbb{N}$, $\mathbb{S}$, and $\mathbb{W}$, respectively. A number of important objects in each experiment are identified by a subscript as follows: for $\mathbb{O} \in \{\mathbb{M}, \mathbb{C}, \mathbb{N}, \mathbb{S}, \mathbb{W}\}$, the input

object, for which the rate-distortion function is approximated by the program, is called $\mathbb{O}_{\mathrm{IN}}$, which is sometimes abbreviated to $\mathbb{O}$. In the denoising experiments, the input object is always constructed by adding noise to an original object. The original object and the noise are called $\mathbb{O}_{\mathrm{ORIG}}$ and $\mathbb{O}_{\mathrm{NOISE}}$, respectively. If Hamming distortion is used, addition is carried out modulo 2, which means that $\mathbb{C}_{\mathrm{IN}}$ equals $\mathbb{C}_{\mathrm{ORIG}}$ XOR $\mathbb{C}_{\mathrm{NOISE}}$. As mentioned before, the search program outputs the reduction of the gene pool, which is the set of considered models. Two important models are also given special names: the model within the gene pool that minimizes the distortion to $\mathbb{O}_{\mathrm{ORIG}}$ constitutes the best denoising of the input object and is therefore called $\mathbb{O}_{\mathrm{BEST}}$, and the minimal sufficient statistic as described in Section 2.2 is called $\mathbb{O}_{\mathrm{MSS}}$. Finally, in the denoising experiments we also give names to the results of the alternative denoising algorithms. Namely, $\mathbb{C}_{\mathrm{NAIVE}}$ is the result of the naive denoising algorithm applied to the noisy cross, $\mathbb{N}_{\mathrm{BLUR}}$ is the convolution of $\mathbb{N}$ with a Gaussian kernel with $\sigma = 0.458$ (which was found to be optimal), and $\mathbb{N}_{\mathrm{JPEG}}$ is the image produced by subjecting $\mathbb{N}$ to JPEG compression at the quality level for which the distortion to $\mathbb{N}_{\mathrm{ORIG}}$ is minimized.

5 Results and Discussion

After running for some time on each input object, our program outputs the reduction of a pool $\mathcal{P}$, which is interpreted as a set of models. For each experiment, we report a number of different properties of these sets. Since we are interested in the rate-distortion properties of the input object $x = \mathbb{O}_{\mathrm{IN}}$, we plot the approximation of the distortion-rate function of each input object: $\tilde{d}_x(r) = \min\{d(x,y) \mid y \in \mathrm{trd}(\mathcal{P}), \tilde{K}(y) \leq r\}$. Such approximations of the distortion-rate function are provided for all experiments. For the grayscale images we also plot the distortion-rate approximation that is achieved by JPEG at different quality levels. Here, the rate is the code length achieved by JPEG, and the distortion is the euclidean distortion to $\mathbb{O}_{\mathrm{IN}}$. We also plot an approximation of the code length function discussed in Section 2.2

$$\tilde{\lambda}_x(r) = \min_{y \in \mathcal{P} \mid \tilde{K}(y) \leq r} \tilde{L}_y(x), \tag{7}$$

minimal sufficient statistics can be identified by locating the minimum of this graph.

5.1 Lossy Compression

5.1.1 Experiment 1: $\mathbb{M}$ouse (Euclidean Distortion)

Our first experiment involved the lossy compression of $\mathbb{M}$, a grayscale image of a mouse. A number of elements of the gene pool are shown in Fig. 3. The pictures show how at low rates, the models capture the most important global structures of the image; at higher rates more subtle properties of the image can be represented. Image (a) shows a rough rendering of the distribution of bright and dark areas in $\mathbb{M}_{\mathrm{IN}}$. These shapes are rectangular, which is probably an artifact of the compression algorithm we used: it is better able to compress images with rectangular structure than with circular structure. There is no real reason why a circular structure should be in any way more complex than a rectangular structure, but most general purpose data compression software is similarly biased. In (b), the rate is high enough that the oval shape of the mouse

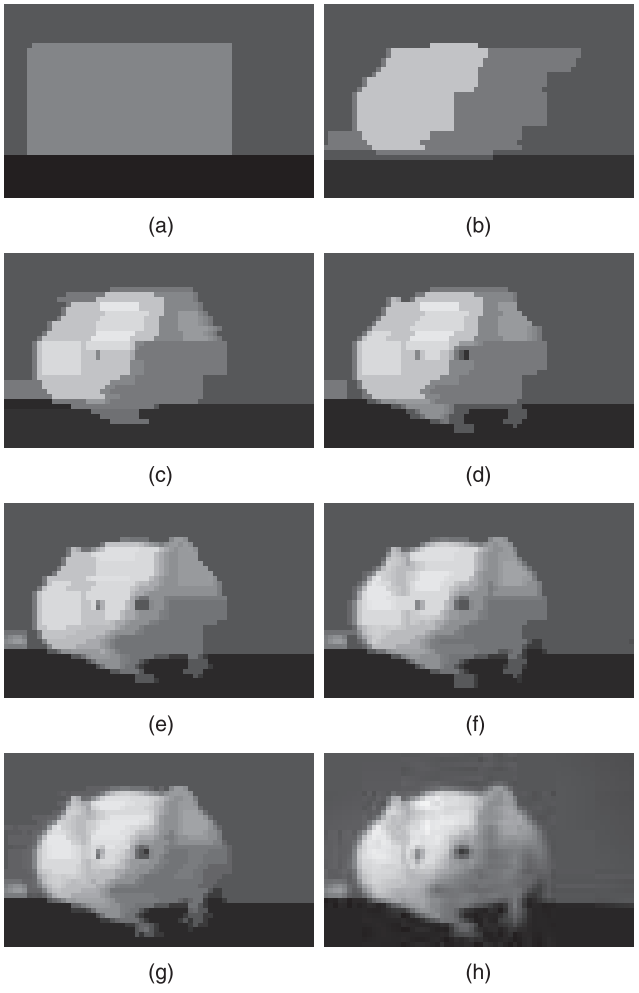


Fig. 3. Lossy image compression results for the mouse (h). The numbers below each image denote its compressed size $\tilde{K}(\cdot)$, total code length $\tilde{L}_{(\cdot)}(\mathbb{M}_{\mathrm{IN}})$ and euclidean distortion $d(\cdot, \mathbb{M}_{\mathrm{IN}})$, respectively. (a) 163.0/19220.9/2210.0. (b) 437.8/17013.6/1080.0. (c) 976.6/15779.7/668.9. (d) 1242.9/15297.9/546.4. (e) 1676.6/14641.9/406.9. (f) 2324.5/14150.1/298.9. (g) 3190.6/13601.4/203.9. (h) 7995.1/7996.1/0.

can be accommodated, and two areas of different overall brightness are identified. After the number of gray shades has been increased a little further in (c), the first hint of the mouse's eyes becomes visible. The eyes are improved and the mouse is given paws in (d). At higher rates, the image becomes more and more refined, but the improvements are subtle and seem of a less qualitative nature.

The code length function (Fig. 4) shows that the only sufficient statistic in the set of models is $\mathbb{M}_{\mathrm{IN}}$ itself, indicating that the image hardly contains any noise. It also shows the rates that correspond to the models that are shown in Fig. 3. By comparing these figures it can be clearly seen that the image quality only starts to deteriorate significantly after more than half of the information in $\mathbb{M}_{\mathrm{IN}}$ has been discarded. Note that this is not a statement about the compression ratio, where the size is related to the size of the *uncompressed* object. For example, $\mathbb{M}_{\mathrm{IN}}$ has an uncompressed size of $64 \cdot 40 \cdot 8 = 20{,}480$ bits, and the representation in Fig. 3b has a compressed size of 3190.6 bits. This representation therefore constitutes compression by a factor of 20,480/3190.6 = 6.42, which is substantial for an image of such small size. At the same

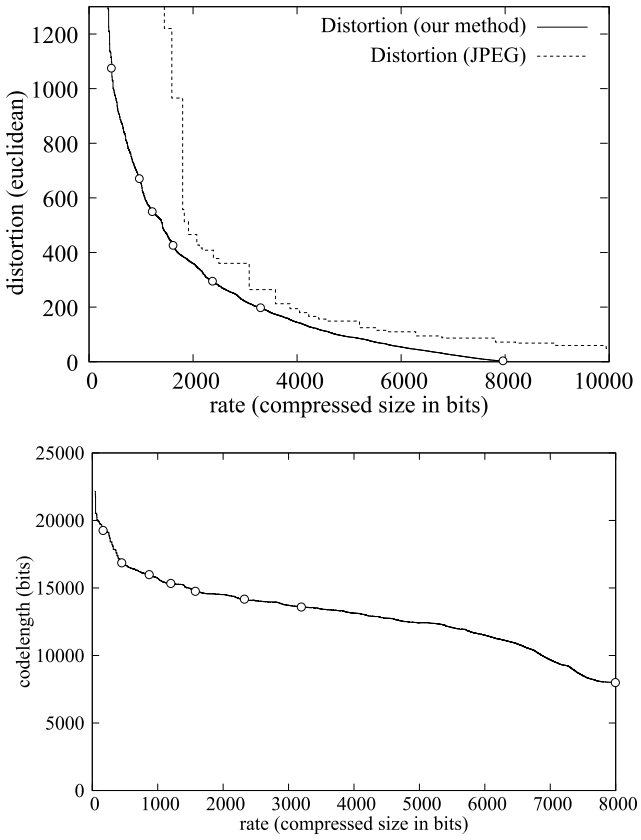


Fig. 4. Approximate distortion-rate and code length functions for the mouse.

time, the amount of *information* is reduced by a factor of $7995.0/3190.6 = 2.51$.

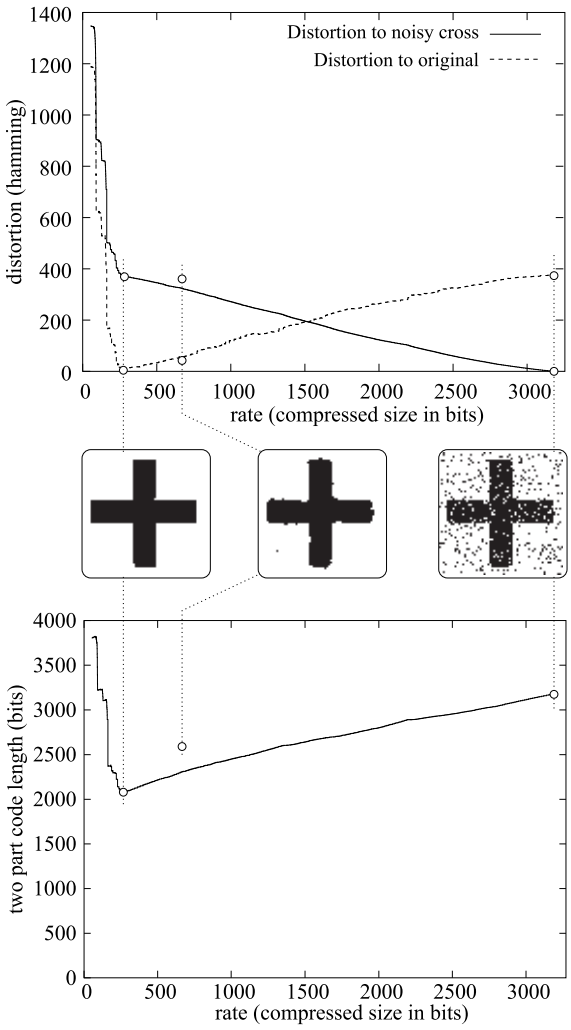
5.2 Denoising

For each denoising experiment, we report a number of important objects, a graph that shows the approximate distortion-rate function and a graph that shows the approximate code length function. In the distortion-rate graph, we plot not only the distortion to $\mathbb{O}_{\mathrm{IN}}$ but also the distortion to $\mathbb{O}_{\mathrm{ORIG}}$, to visualize the denoising success at each rate.

It is important to realize that only the reported minimal sufficient statistic and the results of the naive denoising methods can be obtained without knowledge of the original object—the other objects $\mathbb{O}_{\mathrm{BEST}}$, $\mathbb{O}_{\mathrm{JPEG}}$, and $\mathbb{O}_{\mathrm{BLUR}}$ involve minimizing the distortion to $\mathbb{O}_{\mathrm{ORIG}}$, which cannot be done in practical situations where $\mathbb{O}_{\mathrm{ORIG}}$ is not known.

5.2.1 Experiment 2: Noisy $\mathbb{C}$ross (Hamming Distortion)

In the first denoising experiment, we approximated the distortion-rate function of a monochrome cross $\mathbb{C}_{\mathrm{ORIG}}$ of very low complexity, to which artificial noise was added to obtain $\mathbb{C}_{\mathrm{IN}}$ (the rightmost image in Fig. 5). The best denoising $\mathbb{C}_{\mathrm{BEST}}$ (leftmost image) has a distortion of only three to the original $\mathbb{C}_{\mathrm{ORIG}}$, which shows that the distortion-rate function indeed separates structure and noise extremely well in this example. The bottom graph shows the approximate code length function; the minimum on this graph is the minimal sufficient statistic $\mathbb{C}_{\mathrm{MSS}}$. In this low complexity example, we have $\mathbb{C}_{\mathrm{MSS}} = \mathbb{C}_{\mathrm{BEST}}$, so the best denoising is not only very good in this simple example, but it can also be identified.



	$\mathbb{C}_{\mathrm{BEST}}$=$\mathbb{C}_{\mathrm{MSS}}$	$\mathbb{C}_{\mathrm{NAIVE}}$	$\mathbb{C}_{\mathrm{IN}}$
$\tilde{K}(\cdot)$	260.4	669.2	3178.6
$\tilde{L}_{(\cdot)}(\mathbb{C}_{\mathrm{IN}})$	2081.9	2533.3	3179.6
$d(\cdot, \mathbb{C}_{\mathrm{IN}})$	376	389	0
$d(\cdot, \mathbb{C}_{\mathrm{ORIG}})$	3	40	377

Fig. 5. Denoising a noisy cross. Highlighted objects, from left to right: $\mathbb{C}_{\mathrm{BEST}}$, $\mathbb{C}_{\mathrm{NAIVE}}$, and $\mathbb{C}_{\mathrm{IN}}$. Exact values are in the bottom table.

We did not subject $\mathbb{C}$ to blurring because it is monochrome. Instead we used the extremely simple, "naive" denoising method that is described in Section 4 on this specific image instead. The result is the middle image of Fig. 5; while most of the noise has indeed been removed, 40 errors remain which is a lot more than those incurred by the minimal sufficient statistic. All errors except one are close to the contours of the cross. This illustrates how the naive algorithm is limited by its property that it takes only the local neighborhood of each pixel into account, it cannot represent larger structures such as straight lines.

5.2.2 Experiment 3: $\mathbb{N}$oisy Mouse (Euclidean Distortion)

The noisy mouse poses a significantly harder denoising problem, where the total complexity of the input $\mathbb{N}_{\mathrm{IN}}$ is more than five times as high as for the noisy cross. The results are in Fig. 6. The top-left image (a) is the input object $\mathbb{N}_{\mathrm{IN}}$; it was constructed by adding noise to the original

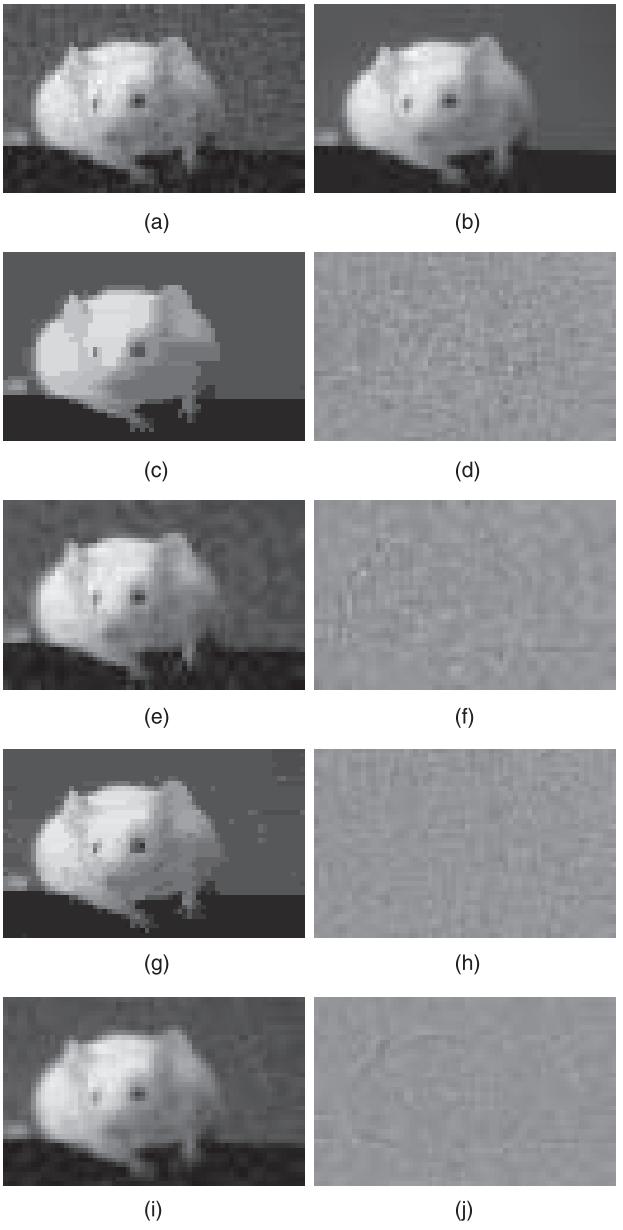


Fig. 6. Denoising results for the noisy mouse (a). The numbers below each image denote its compressed size $\tilde{K}(\cdot)$, total code length $\tilde{L}_{(\cdot)}(\mathbb{N}_{\mathrm{IN}})$, distortion to $\mathbb{N}_{\mathrm{IN}}$ and distortion to $\mathbb{N}_{\mathrm{ORIG}}$, respectively. (a) $\mathbb{N}_{\mathrm{IN}}$, 16699.7/16700.7/0/392.1. (b) $\mathbb{N}_{\mathrm{ORIG}} = \mathbb{M}_{\mathrm{IN}}$. (c) $\mathbb{N}_{\mathrm{MSS}}$, 1969.8/15504.9/474.7/337.0. (d) $\mathbb{N}_{\mathrm{MSS}} - \mathbb{N}_{\mathrm{ORIG}}$. (e) $\mathbb{N}_{\mathrm{JPEG}}$, 3104/16395.4/444.4/379.8. (f) $\mathbb{N}_{\mathrm{JPEG}} - \mathbb{N}_{\mathrm{ORIG}}$. (g) $\mathbb{N}_{\mathrm{BEST}}$, 3354.5/15952.8/368.4/272.2. (h) $\mathbb{N}_{\mathrm{BEST}} - \mathbb{N}_{\mathrm{ORIG}}$. (i) $\mathbb{N}_{\mathrm{BLUR}}(\sigma = 0.458)$, 14117.0/25732.4/260.4/291.2. (j) $\mathbb{N}_{\mathrm{BLUR}} - \mathbb{N}_{\mathrm{ORIG}}$.

noiseless image $\mathbb{N}_{\mathrm{ORIG}} = \mathbb{M}_{\mathrm{IN}}$ (top-right). We display three different denoising results. Image (g) shows $\mathbb{N}_{\mathrm{BEST}}$, the best denoised object from the gene pool. Visually, it appears to resemble $\mathbb{N}_{\mathrm{ORIG}}$ quite well, but it might be the case that there is structure in $\mathbb{N}_{\mathrm{ORIG}}$ that is lost in the denoising process. Because human perception is perhaps the most sensitive detector of structure in image data, we show the difference between $\mathbb{N}_{\mathrm{BEST}}$ and $\mathbb{N}_{\mathrm{ORIG}}$ in (h). We would expect any significant structure in the original image that is lost in the denoising process, as well as structure that is not present in the original image, but is somehow introduced as an artifact of the denoising procedure, to become visible in this residual image. In the case of $\mathbb{N}_{\mathrm{BEST}}$, we cannot make out any particular features in the residual.

We have done the same for the minimal sufficient statistic (c). The result also appears to be a quite successful denoising, although it is clearly of lower complexity than the best one. This is also visible in the residual, which still does not appear to contain much structure, but darker and lighter patches are definitely discernible. Apparently $\mathbb{N}_{\mathrm{IN}}$ does contain some structure beyond what is captured by $\mathbb{N}_{\mathrm{MSS}}$, but this cannot be exploited by the compression algorithm. We think that the fact that the minimal sufficient statistic is of lower complexity than the best possible denoising result should therefore again be attributed to inefficiencies of the compressor.

For comparison, we have also denoised $\mathbb{N}$ using the methods based on blurring and JPEG as described in Section 4. Blurring-based denoising does not actually perform that badly, as shown in (i). The distortion from $\mathbb{N}_{\mathrm{BLUR}}$ to $\mathbb{N}_{\mathrm{ORIG}}$ lies in-between the distortions for $\mathbb{N}_{\mathrm{MSS}}$ and $\mathbb{N}_{\mathrm{BEST}}$, but it differs from those objects in two respects. First, $\mathbb{N}_{\mathrm{BLUR}}$ remains much closer to $\mathbb{N}_{\mathrm{IN}}$, at a distortion of 260.4 instead of 368 or more, and second, $\mathbb{N}_{\mathrm{BLUR}}$ is much less compressible by $\tilde{K}$. These observations are at present not well understood. The contours of the mouse contain high frequency information which is discarded by blurring, creating artifacts that are clearly visible in the residual image (j). Finally, the performance of JPEG is clearly inferior to our method visually as well as in terms of rate and distortion. The result seems to have undergone a smoothing process similar to blurring which introduces similar artifacts in the residual. As before, the comparison may be somewhat unfair because JPEG was not designed for the purpose of denoising, might optimize a different distortion measure and is much faster.

5.2.3 Experiment 4: Regression

In the fourth experiment, we consider a simple regression problem. We start with $\mathbb{S}_{\mathrm{ORIG}}$, a discretization of a sine wave: it is a sequence of 256 bytes, where the ith byte is set to $\lfloor 100 \sin(\pi i/256) \rfloor$ (negative values are represented by byte values 128 through 255). We added mean zero normally distributed noise with variance 14^2 to obtain $\mathbb{S}_{\mathrm{IN}}$, with a euclidean distortion of 209.2.

While block sorting compressors provide an acceptable model for naturally occurring images, they cannot normally compress even the simplest functions very well. For example, a representation of the identity function (in which byte i has value i) cannot be compressed at all. This is because the compressor cannot detect patterns in the input if they have been translated up or down, so a sequence 1, 2, 3, 4 is not matched to a sequence 11,12,13,14 that occurs later in the input. To remedy this, all representations are subjected to a filter before being compressed: from each byte except the first one, the value of the preceding byte is subtracted. Clearly this operation is reversible and thus does not change the information content of a representation. Our example from before, the identity function, is transformed into a sequence of one zero and 255 ones, which can be compressed very well. But in fact any straight line with a "simple" slope (a slope with an easy fraction) can now be compressed quite well. In this manner, the used compressor becomes an interesting model for the description of discrete functions.

The results of the genetic search are given in Fig. 8. In the two graphs on the left hand side, the curved line indicates the original sine wave $\mathbb{S}_{\mathrm{ORIG}}$ and the open circles show $\mathbb{S}_{\mathrm{IN}}$. The small black dots represent the model as found by the program. The top left figure shows that a rate of 40.6 bits suffices to represent the best horizontal straight line, for a distortion to $\mathbb{S}_{\mathrm{ORIG}}$ of 493.4. This is even higher than the distortion for $\mathbb{S}_{\mathrm{IN}}$; thus, naive linear regression does not seem to be a very good approach to this problem! However, the bottom left figure shows the minimal sufficient statistic, which approximates the sine wave with three straight lines. This reduces the distortion to only 60.0 while the rate is still less than 100 bits (compared to a total code length of about 1,600 bits). (We have not plotted $\mathbb{S}_{\mathrm{BEST}}$ as it is very similar to $\mathbb{S}_{\mathrm{MSS}}$, achieving a distortion of 55.5.)

This experiment shows that the used method is a general approach to denoising that does not require many domain specific assumptions, but it also illustrates the impact of approximating the Kolmogorov complexity: in the ideal setting with real Kolmogorov complexity, and given enough data of sufficient precision, a sine wave would be recovered as the best model for the data, because there obviously exists a simple program to compute the sine function. A data compressor cannot compute the sine function, so its use will necessarily result in an approximation, even though with more data that approximation might become more accurate (e.g., consist of more line segments).

5.2.4 Experiment 5: Oscar Wilde Fragment (Edit Distortion)

In the fifth experiment we analyze $\mathbb{W}_{\mathrm{IN}}$, a corrupted quotation from Oscar Wilde. In this case, we have trained the compression algorithm by supplying it with the rest of Chapters 1 and 2 of the same novel as side information, to make it more efficient at compressing fragments of English text. The results are in Fig. 9. We make the following observations regarding the minimal sufficient statistic:

1. In this experiment, $\mathbb{W}_{\mathrm{MSS}} = \mathbb{W}_{\mathrm{BEST}}$ so the minimal sufficient statistic separates structure from noise extremely well here.
2. The distortion is reduced from 68 errors to only 46 errors. 26 errors are corrected ($\triangle$), four are introduced ($\triangledown$), 20 are unchanged ($\bullet$), and 22 are changed incorrectly ($\star$).
3. The errors that are newly introduced ($\triangledown$) and the incorrect changes ($\star$) typically simplify the fragment a lot, in the sense that the compressed size drops significantly. Not surprisingly therefore, many of the symbols marked $\triangledown$ or $\star$ are deletions, or modifications that create a word which is different from the original, but still correct English:

Line	$\mathbb{W}_{\mathrm{ORIG}}$	$\mathbb{W}_{\mathrm{IN}}$	$\mathbb{W}_{\mathrm{MSS}}$
5	or	Nor	of
6	the	Ghe	he
6	any	anL	an
6	learned	JeaFned	yearned
7	course	corze	core
8	then	ehen	when
11	he	fhe	the

Since it would be hard for *any* general-purpose mechanical method (that does not incorporate a sophisticated English language model) to determine that these changes are incorrect, we should not be surprised to find errors of this kind.

Similar experiments on denoising text are also reported in [3]. The described DUDE algorithm does not use side information, but as it can process much larger inputs this becomes less crucial. It would be interesting to see to what extent their results could be improved by taking it into account.

5.2.5 Side Information

The following table shows the compressed size of a number of models for different amounts of side information:

Side information	$\tilde{K}(\mathbb{W}_{\mathrm{ORIG}})$	$\tilde{K}(\mathbb{W}_{\mathrm{MSS}})$	$\tilde{K}(\mathbb{W}_{\mathrm{IN}})$
None	3344.1	3333.7	3834.8
Chapters 1,2 (57kB)	1745.7	1901.9	3234.5
Whole novel (421kB)	1513.6	1876.5	3365.9

Here, $\mathbb{W}_{\mathrm{ORIG}}$ is never included in the side information; also, we do not let $\mathbb{W}_{\mathrm{MSS}}$ vary with side information but keep it fixed at the object reported in Fig. 9c. Clearly, providing side information yields a substantially improved compression performance, and the improvement is typically larger if 1) the amount of side information is larger, or 2) if the compressed object is more similar to the side information. Thus, by giving side information, correct English prose is recognized as "structure" sooner and a better separation between structure and noise is to be expected. The table also shows that if the compressed object is in some way different from the side information, then adding more side information will at some point become counterproductive, presumably because the side information will then cause the compression algorithm to build up false expectations about the object to be compressed, which can be costly.

5.2.6 Sequential Compressors

While denoising performance probably increases if the amount of side information is increased, it was infeasible to do so with our current implementation. Recall from Section 3, that the conditional Kolmogorov complexity $K(y|z)$ is approximated by $\tilde{K}(zy) - \tilde{K}(z)$. The time required to compute this is dominated by the length of z if the amount of side information is much larger than the size of the object to be compressed. This could be remedied by using a compression algorithm that operates sequentially from left to right, because the state of such an algorithm can be cached after processing the side information z; the conditional complexity could then be approximated directly by recalling the state that was cached after processing z, and processing y from there. This also avoids the logarithmic overhead in (4). Many compression algorithms, among which Lempel-Ziv compressors and statistical compressors such as PPM [16] and CTW [17] have this property; our approach could thus be made to work with much more side information by switching to a sequential compressor.

6 Quality of the Approximation

As is clear from the definition, the distortion rate must be a nonincreasing function, and our approximation is also nonincreasing. In every experiment, the gene pool is initialized with $\mathbb{O}_{\mathrm{IN}}$, which always has zero weakness and

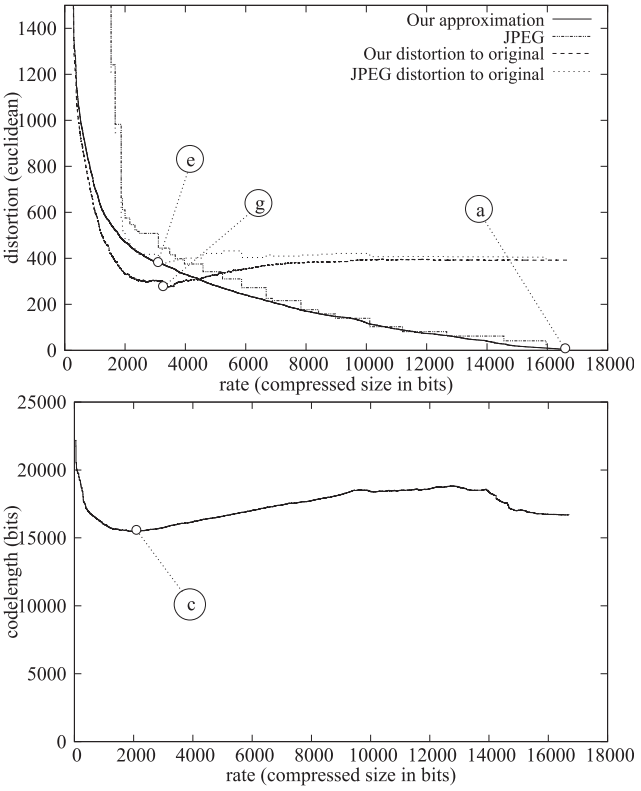


Fig. 7. Approximate distortion-rate and code length functions for the noisy mouse.

must therefore remain in the pool. Therefore, at rate high enough to specify x, the distortion must reach zero.

The shape of the code length function for an object x is more complicated. Let y be the representation for which $d(x,y)=0$, i.e., $y=x$. In theory, the code length can never become less than the complexity of y, and the minimal sufficient statistic witnesses the code length function at the lowest rate at which the code length is equal to the complexity of y. Practically, we found in all denoising experiments that the total code length using the minimal sufficient statistic, $\tilde{L}_{\mathbb{O}_{\mathrm{MSS}}}(\mathbb{O}_{\mathrm{IN}})$, is *less* than the code length $\tilde{K}(\mathbb{O}_{\mathrm{IN}})$ that is obtained by compressing the input object directly. This can be observed in the code length graphs in Figs. 5, 7, 8, and 9. The effect is most clearly visible for the cross, where the separation between structure and noise is most pronounced.

Our hypothesis is that this departure from the theoretical shape of the code length function must be explained by inefficiency of the compression algorithm in dealing with noise. This is evidenced by the fact that it needs 2735.7 bits to encode the noise $\mathbb{C}_{\mathrm{NOISE}}$ that was added to the cross, while only $\log\binom{4096}{377}\approx 1810$ bits would suffice if the noise were specified with a uniform code on the set of indices of all binary sequences with exactly 377 ones out of $64\cdot 64$. Similarly, $\tilde{K}(\mathbb{N}_{\mathrm{NOISE}})=14{,}093$, whereas a literal encoding requires 12,829 or fewer bits (using the bound from the online appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TC.2011.25).

Another strange effect occurs in Figs. 7 and 8: in both, the code length function displays a strange "bump": as the rate is increased beyond the level required to specify the minimal sufficient statistic, the code length goes up as before, but here at very high rates the code length starts dropping again.

It is theoretically possible that the code length function should exhibit such behavior to a limited extent. It can be seen in [6] that a temporary increase in the code length function can occur up to a number of bits that depends on the so-called *covering coefficient*. Loosely speaking this is the density of small distortion balls that is required in order to completely cover a larger distortion ball. The covering coefficient in turn depends on the used distortion function and the number of dimensions. It is quite hard to analyze in the case of euclidean distortion, so we cannot at present say if theory admits such a large increase in the code length function. However, we believe that the explanation is more mundane in this case: we fear that this bump may simply indicate that we interrupted our search procedure too soon. Possibly the bump would disappear altogether if we let our program run for a much longer period of time.

Fig. 4 shows that our approximation of the distortion-rate function is somewhat better than the approximations provided by JPEG, although the difference is not extremely large for higher rates. The probable reason is twofold: on the one hand, we do not know for which distortion function JPEG is optimized, but it might well be something other than euclidean distortion. In that case, the comparison is unfair because our method might perform worse on JPEG's own distortion measure. On the other hand, JPEG is very time efficient: it took only a matter of seconds to compute models at various different quality levels, while it took our own algorithm days or weeks to compute its distortion-rate approximation. Two conclusions can be drawn. On the one hand, if the performance of existing image compression software had been better than the performance of our own method in our experiments, this would have been evidence to suggest that our algorithm does not compute a good approximation to the rate-distortion function. The fact that this is not the case is thus reassuring. Vice versa, if we assume that we have computed a good approximation to the algorithmic rate-distortion function, then our results give a measure of how close JPEG comes to the theoretical optimum; our program can thus be used to provide a basis for the evaluation of the performance of lossy compressors.

7 An MDL Perspective

It is natural to ask how the practical method we described above fits within the established Minimum Description Length theory [18], [19]. After all, MDL was originally developed to obtain a practical version of universal learning based on Kolmogorov complexity, which can even be interpreted as a special case ("ideal MDL," see, e.g., [20], [21]).

We compare the code length function used in MDL to the one we used for algorithmic rate distortion. As stated earlier in (6), the total code length of the source object $x\in\mathcal{X}$ using a representation $y\in\mathcal{Y}$ equals

$$\tilde{K}(y)+L_D(d(x,y)|y)+\log|S_y(d(x,y))|. \tag{8}$$

In this three part code, the first term counts the number of bits required to specify the representation, or hypothesis, for the data and the last term counts the number of bits required to specify the noise. Whether the distortion level of the source

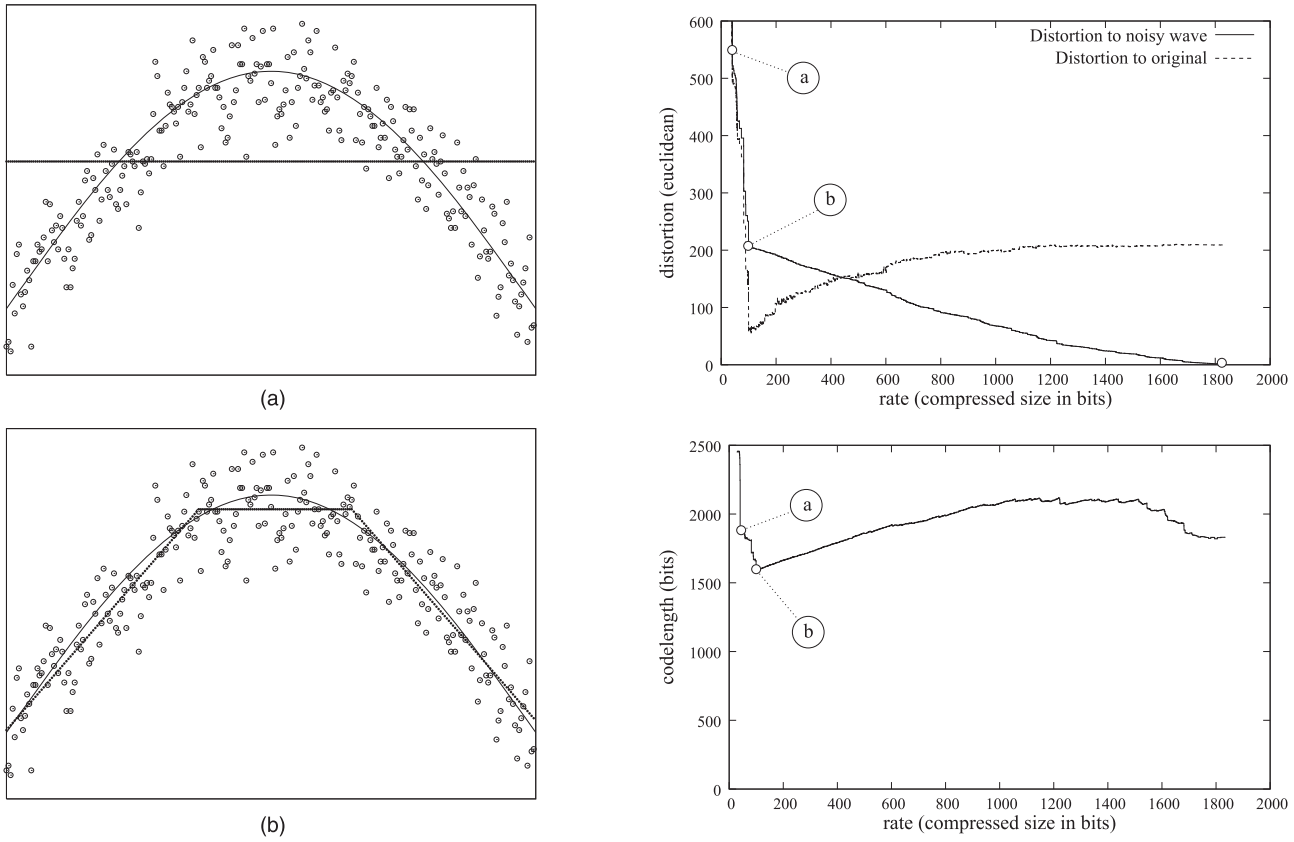


Fig. 8. Regression results. The open circles show the input, closely packed dots show the result. The numbers denote compressed size $\tilde{K}(\cdot)$, total code length $\tilde{L}_{(\cdot)}(\mathbb{S}_{\rm IN})$, distortion to $\mathbb{S}_{\rm IN}$ and distortion to $\mathbb{S}_{\rm ORIG}$, respectively. (a) $40.6/1878.2/522.0/493.4$. (b) $99.8/1592.0/206.6/60.0$.

object given a representation should be interpreted as part of the hypothesis or as noise is debatable; earlier we found it convenient to treat the distortion level as part of the hypothesis, as in (2). But to match algorithmic rate distortion to MDL it is better to think of the distortion level as part of the noise and identify $\mathcal{Y}$ with the available hypotheses.

In the usual description of MDL, see, e.g., [18], the total code length of $x \in \mathcal{X}$ with the help of a hypothesis $y \in \mathcal{Y}$ is

$$L(y) + L(x|y). \tag{9}$$

To explain how algorithmic rate-distortion theory generalizes MDL we will match (8) to (9). Starting with an instance of MDL, specified by the code length functions $L(y)$ and $L(x|y)$, define the probability mass function $P(x|y) = 2^{-L(x|y)}$. Now, the model selected by MDL is equal to a sufficient statistic as found by the rate-distortion algorithm that is obtained by setting $\tilde{K}(y) = L(y)$, $d(x,y) = L(x|y)$, and $L_D(d|y) = -\log P(d(X,y) = d|y)$. Then

$$\begin{aligned} P(x|y) &= P(X = x, L(X|y) = L(x|y)|y) \\ &= P(X = x|y, L(X|y) = L(x|y)) \cdot P(L(X|y) = L(x|y)|y) \\ &= \frac{P(L(X|y) = L(x|y)|y)}{|\{x' \mid L(x'|y) = L(x|y)\}|} \quad = \quad \frac{2^{-L_D(d(x,y)|y)}}{|S_y(d(x,y))|}, \end{aligned}$$

and taking the $-\log$ we find that $L(x|y)$ matches the last two terms of (8): MDL selects a representation that minimizes (8), in other words it selects some sufficient statistic. Note that the only requirement on the distortion function is that for all $y \in \mathcal{Y}$ and all $x, x' \in \mathcal{X}$ it should satisfy $d(x,y) = d(x',y)$ iff $L(x|y) = L(x'|y)$. We chose $d(x,y) = L(x|y)$ for simplicity, and because it allows an interpretation of the distortion as the incurred logarithmic loss.

The correspondence works both ways: starting with a rate-distortion problem specified by some $\tilde{K}$, code L_D and distortion function d, we can also construct the MDL problem by defining $L(y) = \tilde{K}(y)$ and $L(x|y) = L_D(d(x, y)|y) + \log |S_y(d(x,y))|$.

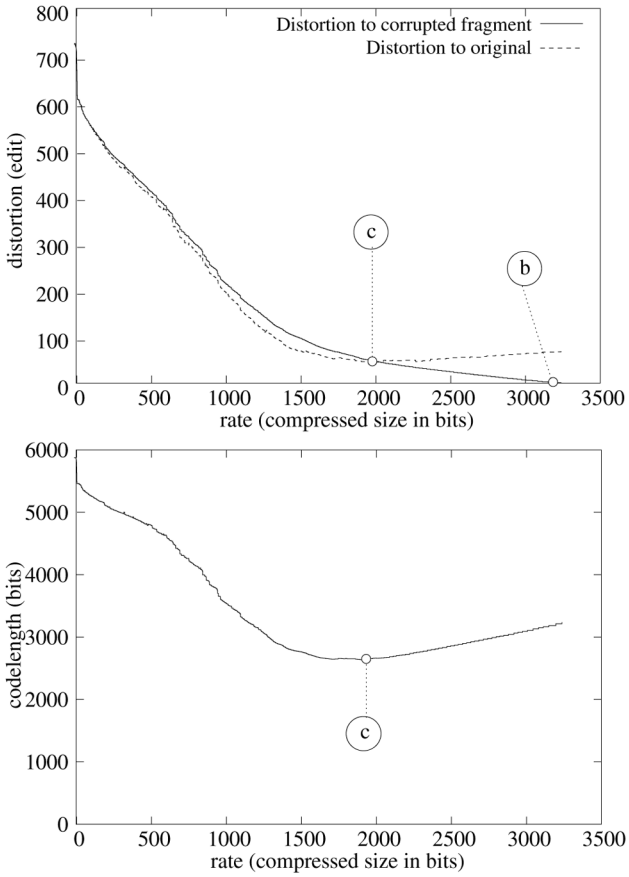
Many descriptions of MDL learning are somewhat unspecific as to *which* hypothesis should be selected if more than one minimize the code length; the issue becomes much clearer in a rate-distortion analysis where one can easily express a preference for the *minimal* sufficient statistic as the best hypothesis for the data. If a thorough analysis of the data is required, it also seems sensible to look at the whole rate-distortion function rather than just at the minimal sufficient statistic, since it provides additional information about the structure that is present in the data at each level of complexity.

8 Conclusion

Algorithmic rate distortion provides a good framework for analysis of large and structured objects. It is based on Kolmogorov complexity, which is not computable. We nevertheless attempted to put this theory into practice by approximating the Kolmogorov complexity of an object by its compressed size. We also generalized the theory to allow side information, which is interpreted as being available to both the sender and the receiver in a transmission over a rate restricted channel. We then described how algorithmic

Beauty, real beauty, ends where an intellectual expression begins. Intellect is in itself a mode of exaggeration, and destroys the harmony of any face. The moment one sits down to think, one becomes all nose, or all forehead, or something horrid. Look at the successful men in any of the learned professions. How perfectly hideous they are! Except, of course, in the Church. But then in the Church they don't think. A bishop keeps on saying at the age of eighty what he was told to say when he was a boy of eighteen, and as a natural consequence he always looks absolutely delightful. Your mysterious young friend, whose name you have never told me, but whose picture really fascinates me, never thinks. I feel quite sure of that.

(a) $\mathbb{W}_{\mathrm{ORIG}}$, the original text



Beauty, real beauty, ends2wheresan intellectual expressoon begins. IntellHct isg in itself a mMde ofSexggeration, an\ destroys theLharmony of n face. :The m1ment one sits down to ahink@ one becomes jll noeˆ Nor all forehbead, or something hNrrid. Look a Ghe successf\l men in anL of te JeaFned professions. How per}ectly tideous 4they re6 Except, of corze, in7 the Ch4rch. BuP ehen in the Church they dol't bthink. =A bishop keeps on saying at the age of eighty what he was told to say wh"n he was aJb4y of eighten, and sja natural cnsequence fhe a(ways looks ab8olstely de[ightfu). Your mysterious youngL friend, wPose name you h\vo never tld me, mut whose picture really fa?scinates Lme,Pnever thinCs. I feel quite surS of that9

(b) $\mathbb{W}_{\mathrm{IN}}$, the corrupted version of the fragment. At 68 randomly selected positions characters have been inserted, deleted or modified. New and replacement characters are drawn uniformly from ASCII symbols 32–126.

Beauty, real beauty, ends-where an intellectual expressoon begins. Intellect is□ in itself a mode of ex□ggeration, and destroys the harmony of □n□ face. □The moment one sits down to think□ one becomes □ll no□e□ □of all forebe□d, or something hirrid. Look a□ □he successf□l men in an□ of t□e yearned provessions. How perfectly tideous □they □re6 Except, of co□r□e, in□ the Charch. But when in the Church they dol't □think. □A□bishop keeps on saying at the age of eighty what he was told to say wh□n he was a□bsy of eight□en, and □sja natural c□nsequence the a□ways looks absolutely de□ightful. Your mysterious young□ friend, whose name you have never t□ld me, mut whose picture really fa□scinates □me, never thinCs. I feel quite sur□ of that9

(c) $\mathbb{W}_{\mathrm{BEST}} = \mathbb{W}_{\mathrm{MSS}}$; it has edit distortion 46 to the original fragment. Marks indicate the error type: ▲=correction; ▼=new error; ●=old error; ★=changed but still wrong. Deletions are represented as □.

Fig. 9. A fragment of *The Picture of Dorian Gray*, by Oscar Wilde.

rate-distortion theory may be applied to lossy compression and denoising problems.

Finding the approximate rate-distortion function of an individual object is a difficult search problem. We describe a genetic algorithm that is very slow, but has the important advantage that it requires only few assumptions about the problem at hand. Judging from our experimental results, our algorithm provides a good approximation, as long as its input object is of reasonably low complexity and is compressible by the used data compressor. The shape of the approximate rate-distortion function, and especially that of the associated three part code length function, is reasonably similar to the shape that we would expect on the basis of theory, but there is a striking difference as well: at rates higher than the complexity of the minimal sufficient statistic, the approximated code length tends to increase with the rate, where theory suggests it should remain constant. We expect that this effect can be attributed to inefficiencies in the compressor.

We find that the algorithm performs quite well in lossy compression, with apparently somewhat better image quality than that achieved by JPEG, although the comparison may not be altogether fair. When applied to denoising, the minimal sufficient statistic tends to slightly underestimate the complexity of the best possible denoising (an example of underfitting). This is presumably again due to inefficiencies in the used compression algorithm.

REFERENCES

[1] C.E. Shannon, "A Mathematical Theory of Communication," *Bell Systems Technical J.,* vol. 27, pp. 379-423, 623-656, 1948.

[2] J. Ziv, "Distortion-Rate Theory for Individual Sequences," *IEEE Trans. Information Theory,* vol. 26, no. 2, pp. 137-143, Mar. 1980.

[3] T. Weissman, E. Ordentlich, G. Seroussi, S. Verdú, and M. Weinberger, "Universal Discrete Denoising: Known Channel," *IEEE Trans. Information Theory,* vol. 51, no. 1, pp. 5-28, Jan. 2005.

[4] S. Jalali and T. Weissman, "Rate-Distortion via Markov Chain Monte Carlo," *Proc. IEEE Int'l Symp. Information Theory (ISIT),* July 2008.

[5] D.L. Donoho, "The Kolmogorov Sampler," technical report, Stanford Univ., Jan. 2002.

[6] N. Vereshchagin and P. Vitanyi, "Rate Distortion and Denoising of Individual Data Using Kolmogorov Complexity," *IEEE Trans. Information Theory,* vol. 56, no. 7, pp. 3438-3454, July 2010.

[7] B.K. Natarajan, "Filtering Random Noise from Deterministic Simals via Data Compression," *IEEE Trans. Signal Processing,* vol. 43, no. 11, pp. 2595-2605, Nov. 1995.

[8] M. Li and P. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications,* third ed. Springer-Verlag, 2008.

[9] R. Cilibrasi and P. Vitányi, "Clustering by Compression," *IEEE Trans. Information Theory.,* vol. 51, no. 4, pp. 1523-1545, Apr. 2005.

[10] M. Burrows and D.J. Wheeler, "A Block-Sorting Lossless Data Compression Algorithm," Digital Equipment Corporation, technical report, Systems Research Center, vol. 124, May 1994.

[11] "Bzip2 and Zzip," www.bzip2.org, debin.net/zzip, lossless Compression Software, 2011.

[12] P. Elias, "Universal Codeword Sets and Representations of the Integers," *IEEE Trans. Information Theory,* vol. IT-21, no. 2, pp. 194-203, Mar. 1975.

[13] V.I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals," *Soviet Physics Doklady,* vol. 10, no. 8, pp. 707-710, 1966.

[14] "Netpbm," www.netpbm.sourceforge.net, open source graphics software, 2011.

[15] G. Chang, B. Yu, and M. Vetterli, "Adaptive Wavelet Thresholding for Image Denoising and Compression," *IEEE Trans. Image Processing,* vol. 9, no. 9, pp. 1532-1546, Sept. 2000.

[16] J.G. Cleary and I.H. Witten, "Data Compression Using Adaptive Coding and Partial String Matching," *IEEE Trans. Comm.,* vol. 32, no. 4, pp. 396-402, Apr. 1984.

[17] F. Willems, Y. Shtarkov, and T. Tjalkens, "The Context-Tree Weighting Method: Basic Properties," *IEEE Trans. Inform. Theory,* vol. 41, no. 3, pp. 653-664, May 1995.

[18] P. Grünwald, *The Minimum Description Length Principle.* MIT Press, June 2007.

[19] J. Rissanen, *Information and Complexity In Statistical Modeling.* Springer, 2009.

[20] P. Vitányi and M. Li, "Minimum Description Length Induction, Bayesianism, and Kolmogorov Complexity," *IEEE Trans. Information Theory,* vol. 46, no. 2, pp. 446-464, Mar. 2000.

[21] S. de Rooij and P.D. Grünwald, "Luckiness and Regret in Minimum Description Length inference," *Handbook for the Philosophy of Statistics,* vol. 7, Chapter VII, 2010.

Steven de Rooij received the PhD degree in 2008 from the University of Amsterdam. Research was done under supervision of Professor Paul Vitányi and Professor Peter Grünwald at CWI, the national research institute for mathematics and computer science in the Netherlands. From 2008 to 2010, he was a research associate in the Statistical Laboratory at the University of Cambridge, before returning to CWI as a postdoctoral researcher. His work focuses on the Minimum Description Length principle, Bayesian inference, model selection, and sequential prediction.

Paul M.B. Vitányi received the PhD degree from the Free University of Amsterdam in 1978. He is a CWI fellow at the national research institute for mathematics and computer science in the Netherlands, and professor of computer science at the University of Amsterdam. He has worked on cellular automata, computational complexity, distributed and parallel computing, machine learning and prediction, physics of computation, Kolmogorov complexity, information theory, quantum computing, publishing about 200 research papers and some books. Together with Ming Li he pioneered applications of Kolmogorov complexity and coauthored *An Introduction to Kolmogorov Complexity and its Applications*, Springer-Verlag, New York, 1993 (3rd edition 2008).